(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 907 125 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
07.04.1999 Bulletin 1999/14

(51) Int Cl.[6]: **G06F 9/44**

(21) Application number: **98306288.6**

(22) Date of filing: **06.08.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **05.09.1997 US 923910**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

(72) Inventors:
- **Percival, Lynn Cleveland**
  Raleigh, North Carolina 27612 (US)
- **Scanlon, John Lane**
  Raleigh, North Carolina 27613 (US)

(74) Representative: **Zerbi, Guido Maria**
Intellectual Property Department,
IBM United Kingdom Ltd.,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Method and system for automatically detecting and selecting updated versions of a set of files

(57) A list of various versions of file sets that need to be reconciled (collisions) are automatically loaded into a collision handler. The user works through the list, resolving differences by selecting one of the files or by merging several of them through selection and modification, as required. The user is provided with buttons that allow rapid changes between different styles of views (Split, Composite, Split-Merge, Composite-Merge) which allows the user to see the files in a number of different ways to further facilitate the process.

EP 0 907 125 A2

Printed by Jouve, 75001 PARIS (FR)

Description

Field of the Invention

[0001] This invention relates in general to computer software, and in particular to a method and system for automatically notifying a user that a file has been changed/updated by several different persons at the same time in order to allow the various versions to be reconciled.

Background of the Invention

[0002] In a programming environment, programmers may check out a file to write new programming, to change/update and/or debug. With the use of library type storage systems, this may be done by several different programmers simultaneously. Thus, if each person makes one or more changes, the others may not know what has been done. When the files are returned to the library from the various programmers, there will be no single correct and updated version.

[0003] Currently there are tools that allow a user to merge one set of changed files at a time, however, the user is required to determine how many sets there are and work through them keeping track of changes manually. Obviously, this is time consuming and prone to error. Other tools may automatically determine all the relevant sets of files that need to be updated, but the user must still work through these sets manually.

[0004] Thus, there is a need for a method and system for automatically selecting updated versions of a file and resolving conflicts therein.

Summary of the Invention

[0005] According to the present invention we provide a method of merging a plurality of various versions of a set of computer files, comprising the steps of: automatically loading each of the plurality of various versions of the set of computer files into a collision handler; displaying said versions with said collision handler; and resolving any differences between said versions by graphically selecting a preferred one of said versions.

[0006] Further according to the present invention we provide a system for merging a plurality of various versions of a set of computer files, comprising: means for automatically loading each of the plurality of various versions of the set of computer files into a collision handler; means for displaying said versions with said collision handler; and means for resolving any differences between said versions by graphically selecting a preferred one of said versions.

[0007] Also according to the present invention we provide a computer program product recorded on computer readable medium for merging a plurality of various versions of a set of computer files, comprising: computer readable means for automatically loading each of the plurality of various versions of the set of computer files into a collision handler; computer readable means for displaying said versions with said collision handler; and computer readable means for resolving any differences between said versions by graphically selecting a preferred one of said versions.

[0008] In accordance with one embodiment of the present invention, a plurality of various versions of a computer file are merged. Each of the plurality of various versions are automatically loaded into a collision handler. The various versions are then displayed to the user with the collision handler. The user may then resolve any differences between the various versions by visually selecting a preferred one of the versions.

[0009] The versions may be displayed to the user in any of a plurality of views in accordance with the users preference. The views are: split, composite, split-merge and composite-merge.

[0010] The solution as provided according to a preferred embodiment of the present invention takes a list of file sets that have to be reconciled (collisions) and loads them into the collision handler. Then the user works through the list, resolving the differences by selecting one of the files or by merging several of them through selection and modification, as required. The user is provided with buttons that allow them to rapidly change between views (Split, Composite, Split-Merge, Composite-Merge). This allows the user to see the files in a number of different ways to further facilitate the process.

[0011] These and other features, aspects, and advantages of the present invention will become better understood with reference to the following description, appended claims, and accompanying drawings.

Brief Description of the Drawings

[0012] For a more complete understanding of the present invention and for further advantages thereof, reference is now made to the following Detailed Description taken in conjunction with the accompanying drawings, in which:

Figure 1 is a pictorial representation of a data processing system which may be utilized to implement a method and system of the present invention;

Figure 2 shows one scenario whereby several users modify a file in parallel in accordance with the present invention;

Figure 3 illustrates a collision handler in a composite-merge view in accordance with the present invention;

Figure 4 illustrates a collision handler in a split-merge view in accordance with the present invention;

Figure 5 illustrates a collision handler in a composite view in accordance with the present invention;

Figure 6 illustrates a collision handler in a split view in accordance with the present invention;

Figure 7 illustrates a collision handler with direct insertion of material into a merge target in accordance with the present invention;

Figure 8 illustrates a collision handler showing all differences resolved in accordance with the present invention;

Figure 9 illustrates a collision handler showing selection of merged target for resolving collision in accordance with the present invention; and

Figure 10 illustrates a high level flowchart of the present invention.

Detailed Description of the Invention

[0013] Referring to Figure 1, there is depicted a graphical representation of a data processing system 8, which may be utilized to implement the present invention. As may be seen, data processing system 8 may include a plurality of networks, such as Local Area Networks (LAN) 10 and 32, each of which preferably includes a plurality of individual computers 12 and 30, respectively. Of course, those skilled in the art will appreciate that a plurality of Intelligent Work Stations (IWS) coupled to a host processor may be utilized for each such network. Each said network may also consist of a plurality of processors coupled via a communications medium, such as shared memory, shared storage, or an interconnection network. As is common in such data processing systems, each individual computer may be coupled to a storage device 14 and/or a printer/output device 16 and may be provided with a pointing device such as a mouse 17.

[0014] The data processing system 8 may also include multiple mainframe computers, such as mainframe computer 18, which may be preferably coupled to LAN 10 by means of communications link 22. The mainframe computer 18 may also be coupled to a storage device 20 which may serve as remote storage for LAN 10. Similarly, LAN 10 may be coupled via communications link 24 through a sub-system control unit/communications controller 26 and communications link 34 to a gateway server 28. The gateway server 28 is preferably an IWS which serves to link LAN 32 to LAN 10.

[0015] With respect to LAN 32 and LAN 10, a plurality of documents or resource objects may be stored within storage device 20 and controlled by mainframe computer 18, as resource manager or library service for the resource objects thus stored. Of course, those skilled in the art will appreciate that mainframe computer 18 may be located a great geographic distance from LAN 10 and similarly, LAN 10 may be located a substantial distance from LAN 32. For example, LAN 32 may be located in California while LAN 10 may be located within North Carolina and mainframe computer 18 may be located in New York.

[0016] Software program code which employs the present invention is typically stored in the memory of a storage device 14 of a stand alone workstation or LAN server from which a developer may access the code for distribution purposes, the software program code may be embodied on any of a variety of known media for use with a data processing system such as a diskette or CD-ROM or may be distributed to users from a memory of one computer system over a network of some type to other computer systems for use by users of such other systems. Such techniques and methods for embodying software code on media and/or distributing software code are well-known and will not be further discussed herein.

[0017] To illustrate the present invention, a case of three developers (User A, User B, and User C) working concurrently along a time line with a common pool of files will be used as shown in Figure 2). After creating a Workarea for each user in blocks 100, 200 and 300, respectively, Users A, B and C have each checked out files in blocks 102, 202, and 302, respectively. One of the files, "Newfm.c," has been checked out by both User A and User B. Further, "Newfm.c" has been modified by both Users A and B at blocks 104 and 204, respectively. User B checks the file "Newfm.c" back in first at block 206, so no collision is detected at block 208. However, when User A attempts to check file "Newfm.c" in at block 106, the version in the database, which now contains User B's changes, conflicts with User A's version, and a collision is detected at block 108.

[0018] Similarly, Users A and C both have checked out and modified "Globvar.c" in blocks 102, 104 and 302, 304, respectively. Again, since User C checks "Globvar.c" back in first at block 306, no collision is detected at block 308. When User A attempts to check "Globvar.c" in at block 106, a collision results at block 108. User A now must deal with two collisions at block 110.

[0019] In contrast, "file8.c," which has been checked out by both Users B and C, has been modified only by User B. User C checks it back in at block 306 without having made any changes and, being first back in, creates no collision at block 308. When User B checks "file8.c" back in with changes at block 206, no collision is detected at block 208 since User C had not modified the file.

[0020] Following is a description of how User A would deal with the two collisions, first with a prior art collision handling procedure; next with a prior art external file merge tool; and finally with the present invention, referred to herein as the Multiple Collision Handler (MCH). It is to be understood that dealing with only two collisions is an extremely simple example and is used only to il-

lustrate the concepts. In a large-scale development effort, with numerous developers and files, multiple collisions may result when a developer attempts to check in a large set of files. In a preferred embodiment, various colours would be used to designate the various versions being reconciled. However, due to drawing constraints, greyed out areas will be identified by cross-hatching, a first version will be identified by slashing, and a second version will be identified by reverse slashing.

[0021] In the first example using a prior art collision handling procedure, once collisions have been detected, User A is provided with a message that collisions have occurred. The user must navigate a menubar (i.e., "Parts" --> "Collision records...") to display a dialogue with a list of collision records, select a collision record, and either specify which version of the file to use or specify that the files are to be merged. If "Merge" is specified, a window appears with both files displayed and their differences are highlighted. (Note: Even if the user does not intend to actually merge the files, the merge option is the only way to compare the files to determine which one to actually check in. He would then renavigate the menubar to reselect that collision and specify which file to use.) To merge the files, User A must mark blocks or lines of code to be excluded from the target (i.e., the file which will be checked into the database), and then saves and checks in the resulting file. To deal with the second collision, User A must start all over again, once again navigating the menubar ("Parts" -- > "Collision records..."), selecting another collision record from the list, etc. This process must be repeated for each of the detected collisions, i.e., for each file pair with differences. Thus, this prior art procedure is too slow.

[0022] With a prior art external tool, User A would have been notified that collisions exist, and would still have to navigate a menubar to display the list of collisions. To compare the files creating a collision, the user would then invoke the external tool, open the local file (i.e., the file he/she is attempting to check in), locate the database version, and open that version within the external tool. At this point, User A can decide to use one of the two versions without merging and by returning to the list of collisions, he/she can select which version to use. To merge the files, the user must select which version of each highlighted difference to use for the target. This merged file can then be saved as the new local file, and when User A attempts to check it in, another collision will occur. This time, the user would navigate to the collision list again, select the new collision, and select the newly merged version as the file to use. (Note: The original collision would first have had to be cleared from the list; this could be accomplished by selecting the original collision and selecting the database version. However, this must be done before checking in the merged version.) Again, this method is too slow and is also too complex.

[0023] The Multiple Collision Handler (MCH) in accordance with the present invention avoids all of the repeated menubar navigation, and all of the explicit file manipulation found in the prior art methods. When collisions are detected, MCH is automatically invoked. Figure 3 shows the Multiple Collision Handler, generally identified by reference numeral 320, in an initial state. In an upper left hand pane 322, Collision Records generated by a work area integration are listed. In this case there are collisions on file "Newfm.c," identified by reference numeral 324 and on file "Globvar.c," identified by reference numeral 326. The first collision (file 324) is selected by default. In an upper right hand pane 328, information is listed relating to the first collision, e.g., the number of differences, who the part was last checked in by and when.

[0024] Buttons below the upper pane control the views that the user may utilize to resolve the conflicts. Split View button 330 will display a side-by-side view of the different part versions; Composite View button 332 will display an interleaved view; Split-Merge View button 334 will display an over-under view of the part versions beside the Merge work product or target; and Composite-Merge View button 336 will display an interleaved view of the parts beside the Merge Target. A Composite-Merge view is shown as selected in Figure 3.

[0025] A "Next Difference" button 340 and a "Previous Difference" button 342 allow the user to move through the detected differences in the part versions. An "Edit" button 344 allows the user to edit a selected line in the Merge Target, and a "Use Merge target" button 346 becomes active when all differences have been resolved.

[0026] Figure 4 shows a Split-Merge View, generally indicated by reference numeral 400. The two part/file versions are shown in an over-under view with the Merge Target beside them. Figure 5 shows a Composite View, generally identified by reference numeral 500, in which part versions are shown interleaved without a Merge Target. Buttons 502 and 504 allow the user to select one of the versions (Database or Local). Figure 6 shows a Split View, generally indicated by reference numeral 600, with the versions shown side-by-side and buttons 602 and 604 available to select a version.

[0027] Figure 7 again shows a Composite-Merge View, generally identified by reference numeral 700, with an item 702 (line 14) already added to the Merge-Target 704. After the user selects the desired version to resolve the difference, the Merge Target is automatically filled-in and the view is advanced to the next difference. The source is a specific colour (shown as hatching in the instant example) which is carried to the target. Continuing the example of Figure 7, Figure 8 shows that all differences have been resolved, the origin of the items in the Merge Target 704, and the activation of a button 806 to select the Merge Target to resolve the collision. Additionally, Figure 9 shows the condition after the user has selected the Use Merge Target button 806. An icon 902 in the Collision records pane 904 and an icon 906 in the Description pane 908 have changed to show that the Merge Target will be used to resolve that collision.

The user is then ready to work on the next collision.

[0028] The flowchart in Figure 10 provides an overview of the flow of the present invention. The present invention starts at block 940 when the user attempts to integrate a Workarea where changes have been made. The system then checks for collisions at block 942, and it is determined at decision block 944 whether or not a collision is detected. If the response to decision block 944 is no, the present invention ends at 946. If the response to decision block 944 is yes, (collisions are detected) the system warns the user and requests direction from the user at block 948.

[0029] It is then determined at decision block 950 whether or not the user wants to reconcile the differences. If the response to decision block 950 is no, the present invention exits at 952. If the response to decision block 950 is yes, (the user decides to reconcile the collisions) the system populates the list of collision records, makes temporary copies of the files, and analyzes the files for differences at block 954. At block 956 the lists are displayed in the default view. The user modifies the view as needed to decide whether to use the database or local version as-is, or merge the two versions at block 958.

[0030] It is then determined at decision block 960 whether or not the user has decided to merge the versions. If the response to decision block 960 is no (the user selected use Local or DataBase version), the collision record is marked as being resolved and the icon changed to show the origin of the version as appropriate, and the present invention proceeds to decision block 962, as will be subsequently described in greater detail.

[0031] If the response to decision block 960 is yes (the user is merging the versions), the system highlights the differences and places selection emphasis on the next one at block 964. The user selects the DataBase or Local version of that difference at block 966, and the system places it into the Merge Target, preserving the colour to show the origin at block 968. The user may then edit the line, if desired, by selecting the Edit button. It is then determined at decision block 970 whether or not this was the last difference to be resolved. If the response to decision block 970 is no, the present invention returns to block 964, as previously described above.

[0032] If the response to decision block 970 is yes, the system activates the Use Merge Target button at block 972. When the user selects the Use Merge Target button at block 974, the collision record is marked as being resolved at block 976, and the icon is changed to indicate that the resolution was through a merge. Subsequent to block 976 or decision block 960, it is determined at decision block 962 whether or not this is the last collision. If the response to decision block 962 is no, the present invention returns to block 956, as previously described above. If the response to decision block 962 is yes, it is determined at decision block 978 wether or not to integrate the Workarea. If the response to decision block 978 is no, the present invention ends at 946. If the response to decision block 978 is yes, the system integrates the Workarea into the code base at block 980 and ends at 946.

[0033] As a result of the present invention, the problem of merging several sets of files in a programming environment is solved. Prior to the present invention, available tools would only allow the user to merge one set of files at a time. The user was required to determine how many sets there were and to work through them, keeping track manually. This was time consuming and error prone.

[0034] The solution as provided herein is to take a list of file sets that have to be reconciled (collisions) and load them into the collision handler. Then the user works through the list, resolving the differences by selecting one of the files or by merging several of them through selection and modification, as required. The user is provided with buttons that allow them to rapidly change between views (Split, Composite, Split-Merge, Composite-Merge). This allows the user to see the files in a number of different ways to further facilitate the process.

Claims

1. A method of merging a plurality of various versions of a set of computer files, comprising the steps of:

   automatically loading each of the plurality of various versions of the set of computer files into a collision handler;
   displaying said versions with said collision handler; and
   resolving any differences between said versions by graphically selecting a preferred one of said versions.

2. The method of Claim 1, wherein said step of displaying further comprises:

   displaying said versions in a split view.

3. The method of Claim 1, wherein said step of displaying further comprises:

   displaying said versions in a composite view.

4. The method of Claim 1, wherein said step of displaying further comprises:

   displaying said versions in a split-merge view.

5. The method of Claim 1, wherein said step of displaying further comprises:

   displaying said versions in a composite-merge view.

6. A system for merging a plurality of various versions of a set of computer files, comprising:

means for automatically loading each of the plurality of various versions of the set of computer files into a collision handler;
means for displaying said versions with said collision handler; and
means for resolving any differences between said versions by graphically selecting a preferred one of said versions.

7. The system of Claim 6, wherein said means for displaying further comprises:

means for displaying said versions in a split view.

8. The system of Claim 6, wherein said means for displaying further comprises:

means for displaying said versions in a composite view.

9. The system of Claim 6, wherein said means for displaying further comprises:

means for displaying said versions in a split-merge view.

10. The system of Claim 4, wherein said means for displaying further comprises:

means for displaying said versions in a composite-merge view.

11. A computer program product recorded on computer readable medium for merging a plurality of various versions of a set of computer files, comprising:

computer readable means for automatically loading each of the plurality of various versions of the set of computer files into a collision handler;
computer readable means for displaying said versions with said collision handler; and
computer readable means for resolving any differences between said versions by graphically selecting a preferred one of said versions.

12. The program product of Claim 11, wherein said computer readable means for displaying further comprises:

computer readable means for displaying said versions in a split view.

13. The program product of Claim 11, wherein said computer readable means for displaying further comprises:

computer readable means for displaying said versions in a composite view.

14. The program product of Claim 11, wherein said computer readable means for displaying further comprises:

computer readable means for displaying said versions in a split-merge view.

15. The program product of Claim 11, wherein said computer readable means for displaying further comprises:

computer readable means for displaying said versions in a composite-merge view.

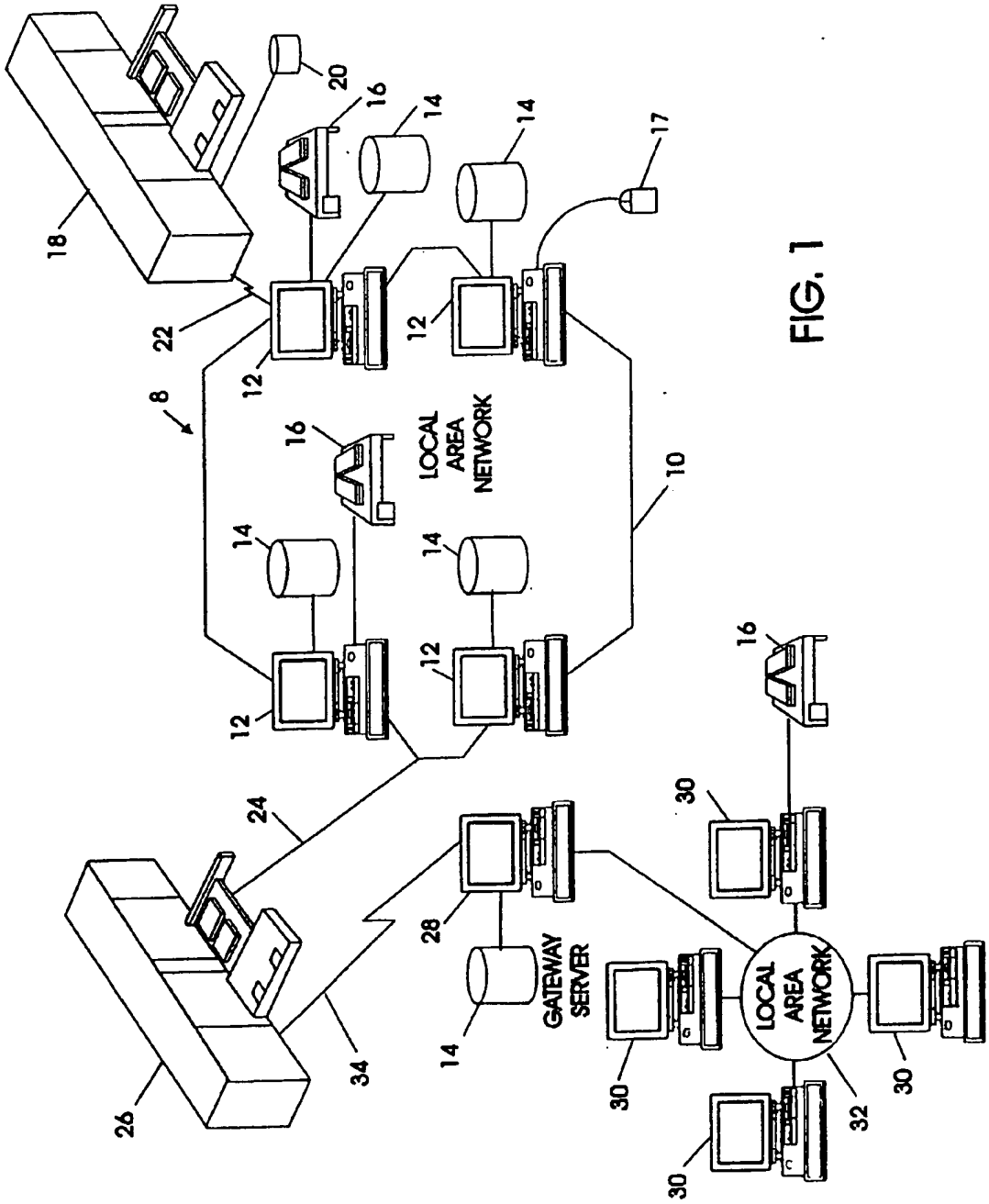
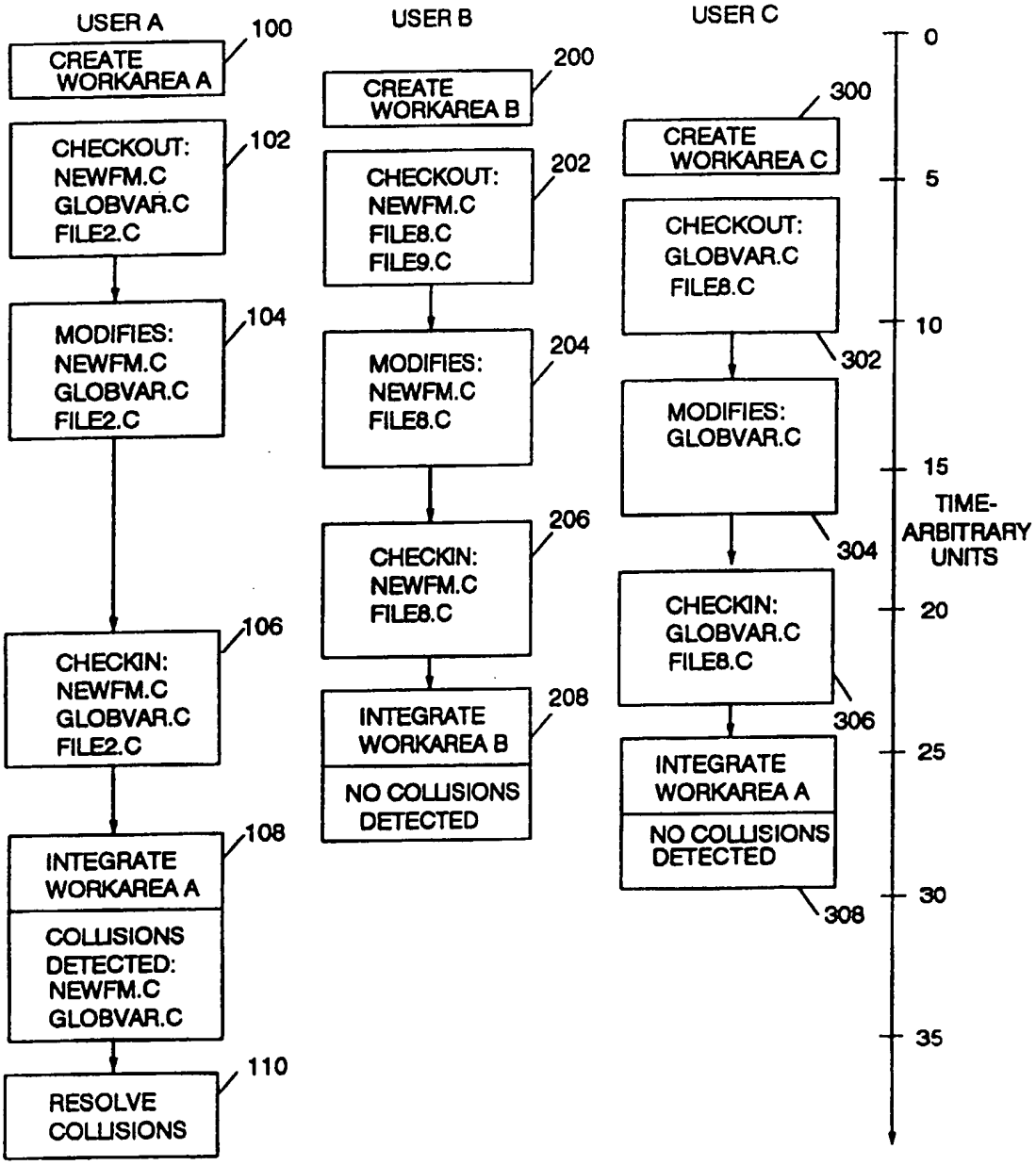

18
20
16
14
14
17
22
12
12
8
LOCAL AREA NETWORK
16
10
14
14
12
12
16
24
30
28
GATEWAY SERVER
LOCAL AREA NETWORK
14
30
30
30
32
26
34

FIG. 1

FIG. 2


USER A
100
CREATE WORKAREA A
102
CHECKOUT: NEWFM.C GLOBVAR.C FILE2.C
104
MODIFIES: NEWFM.C GLOBVAR.C FILE2.C
106
CHECKIN: NEWFM.C GLOBVAR.C FILE2.C
108
INTEGRATE WORKAREA A
COLLISIONS DETECTED: NEWFM.C GLOBVAR.C
110
RESOLVE COLLISIONS
USER B
200
CREATE WORKAREA B
202
CHECKOUT: NEWFM.C FILE8.C FILE9.C
204
MODIFIES: NEWFM.C FILE8.C
206
CHECKIN: NEWFM.C FILE8.C
208
INTEGRATE WORKAREA B
NO COLLISIONS DETECTED
USER C
300
CREATE WORKAREA C
CHECKOUT: GLOBVAR.C FILE8.C
302
MODIFIES: GLOBVAR.C
304
CHECKIN: GLOBVAR.C FILE8.C
306
INTEGRATE WORKAREA A
NO COLLISIONS DETECTED
308
0
5
10
15
20
25
30
35
TIME-ARBITRARY UNITS

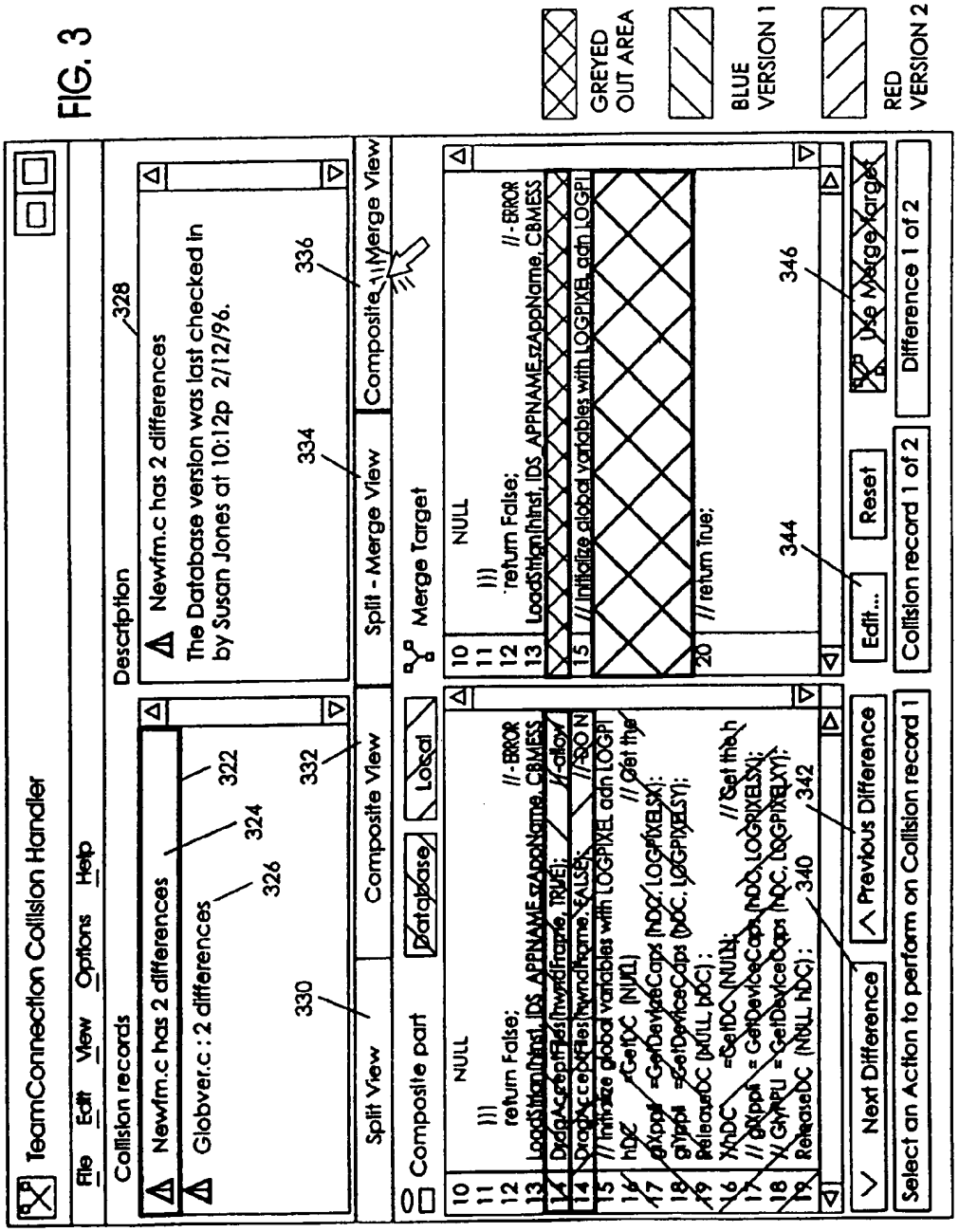

320
TeamConnection Collision Handler
File Edit View Options Help
Collision records
Description
328
Newfm.c has 2 differences
Globver.c : 2 differences
322
324
326
330
332
334
336
Newfm.c has 2 differences
The Database version was last checked in by Susan Jones at 10:12p 2/12/96.
Split View
Composite View
Split - Merge View
Composite View
Merge Target
Composite part
Database
Local
Next Difference
Previous Difference
Edit...
Reset
Use Merge Target
Collision record 1 of 2
Difference 1 of 2
Select an Action to perform on Collision record 1
340
342
344
346
GREYED OUT AREA
BLUE VERSION 1
RED VERSION 2

FIG. 3

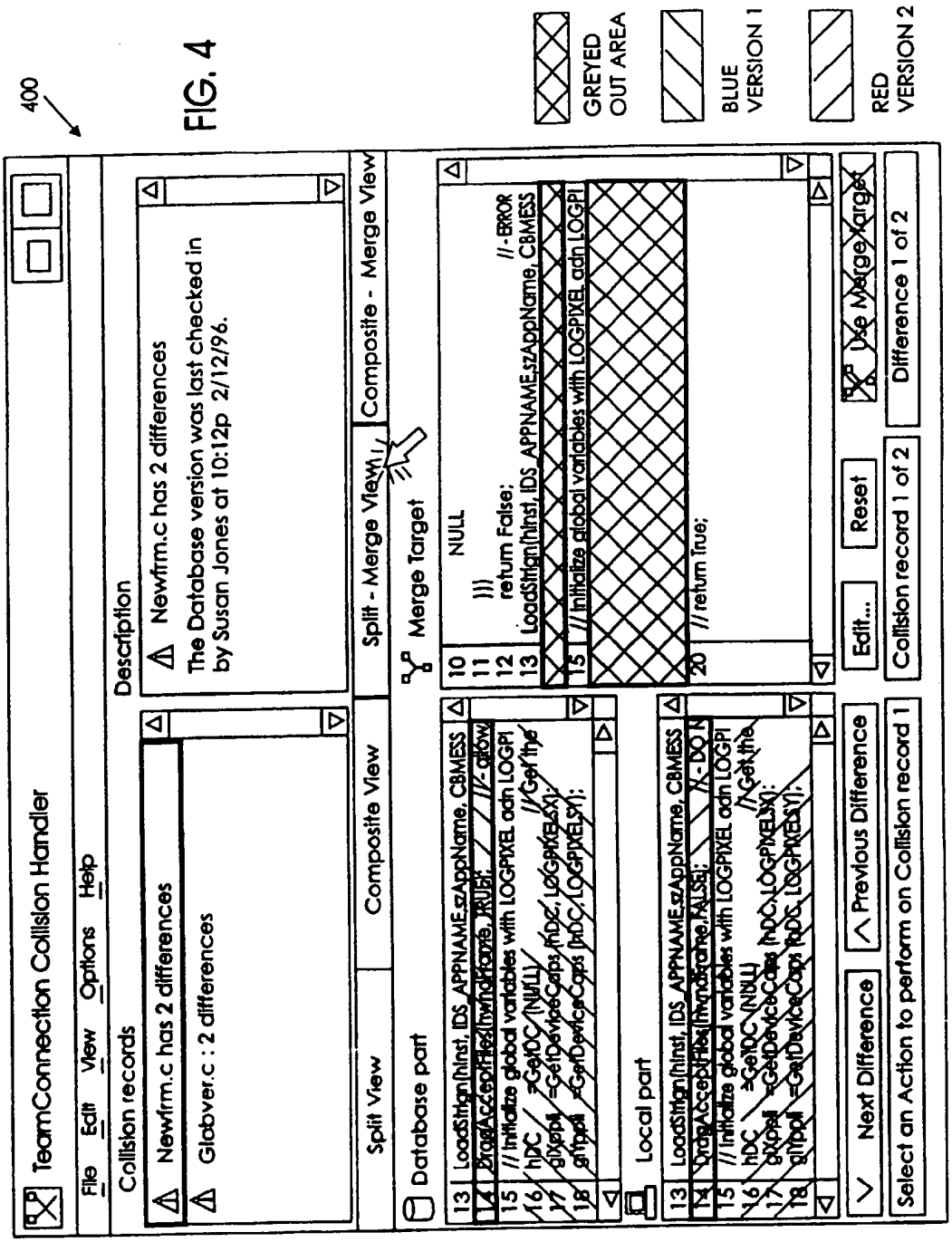

400
TeamConnection Collision Handler
File Edit View Options Help
Collision records
Newfrm.c has 2 differences
Globver.c : 2 differences
Description
Newfrm.c has 2 differences
The Database version was last checked in by Susan Jones at 10:12p 2/12/96.
Split View
Composite View
Split - Merge View
Composite - Merge View
Database part
Merge Target
Local part
Next Difference
Previous Difference
Edit...
Reset
Use Merge Target
Select an Action to perform on Collision record 1
Collision record 1 of 2
Difference 1 of 2
GREYED OUT AREA
BLUE VERSION 1
RED VERSION 2

FIG. 4

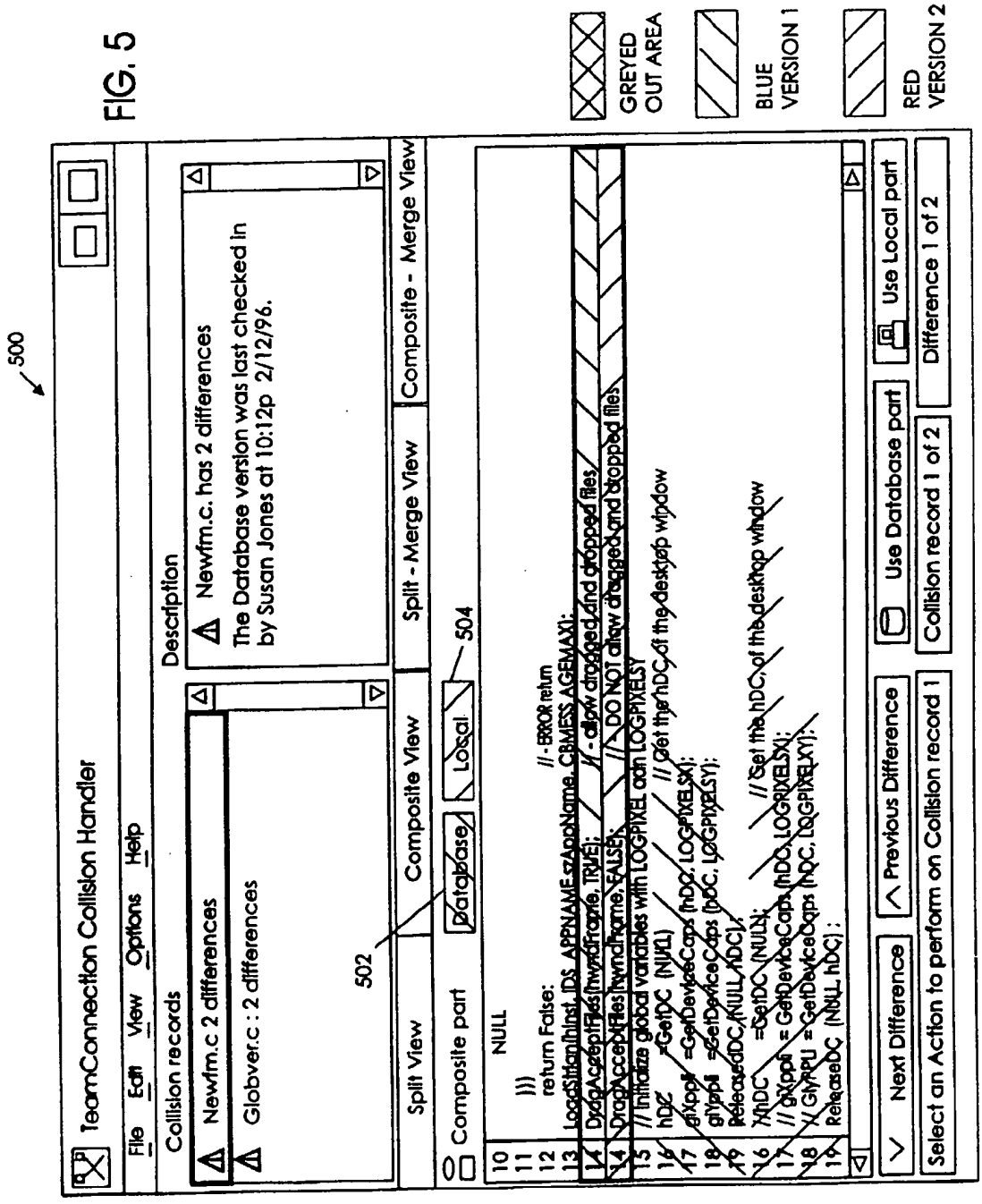

500
TeamConnection Collision Handler
File Edit View Options Help
Collision records
Newfm.c 2 differences
Globver.c : 2 differences
Description
Newfm.c. has 2 differences
The Database version was last checked in by Susan Jones at 10:12p 2/12/96.
502
Split View
Composite View
Split - Merge View
Composite - Merge View
Composite part
Database
Local
504
NULL
Next Difference
Previous Difference
Use Database part
Use Local part
Select an Action to perform on Collision record 1
Collision record 1 of 2
Difference 1 of 2
GREYED OUT AREA
BLUE VERSION 1
RED VERSION 2

FIG. 5

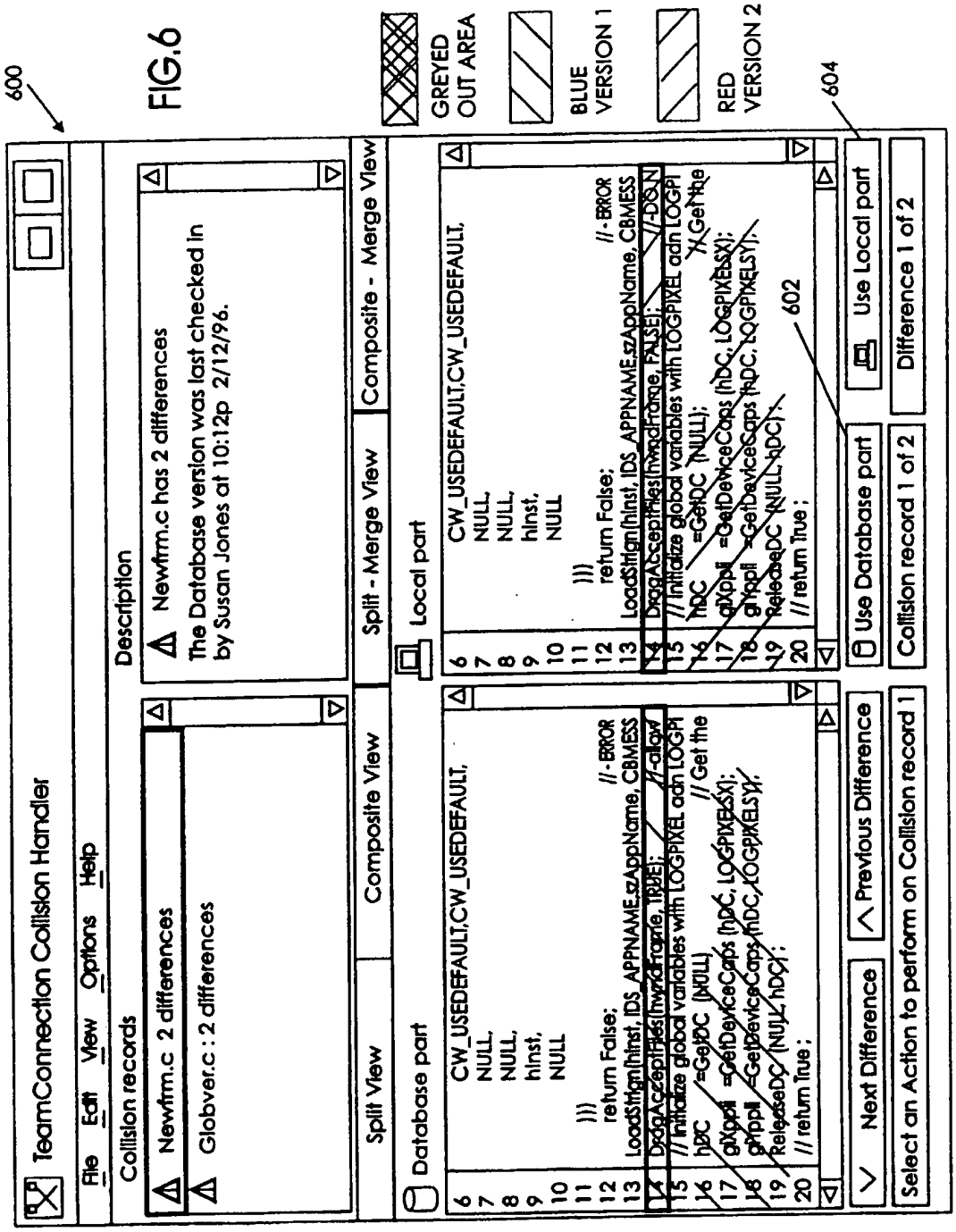

600
TeamConnection Collision Handler
File Edit View Options Help
Collision records
Newfrm.c 2 differences
Globver.c : 2 differences
Description
Newfrm.c has 2 differences
The Database version was last checked in by Susan Jones at 10:12p 2/12/96.
Split View
Composite View
Split - Merge View
Composite - Merge View
Database part
Local part
Next Difference
Previous Difference
Use Database part
Use Local part
Select an Action to perform on Collision record 1
Collision record 1 of 2
Difference 1 of 2
602
604
GREYED OUT AREA
BLUE VERSION 1
RED VERSION 2

FIG.6

FIG. 7

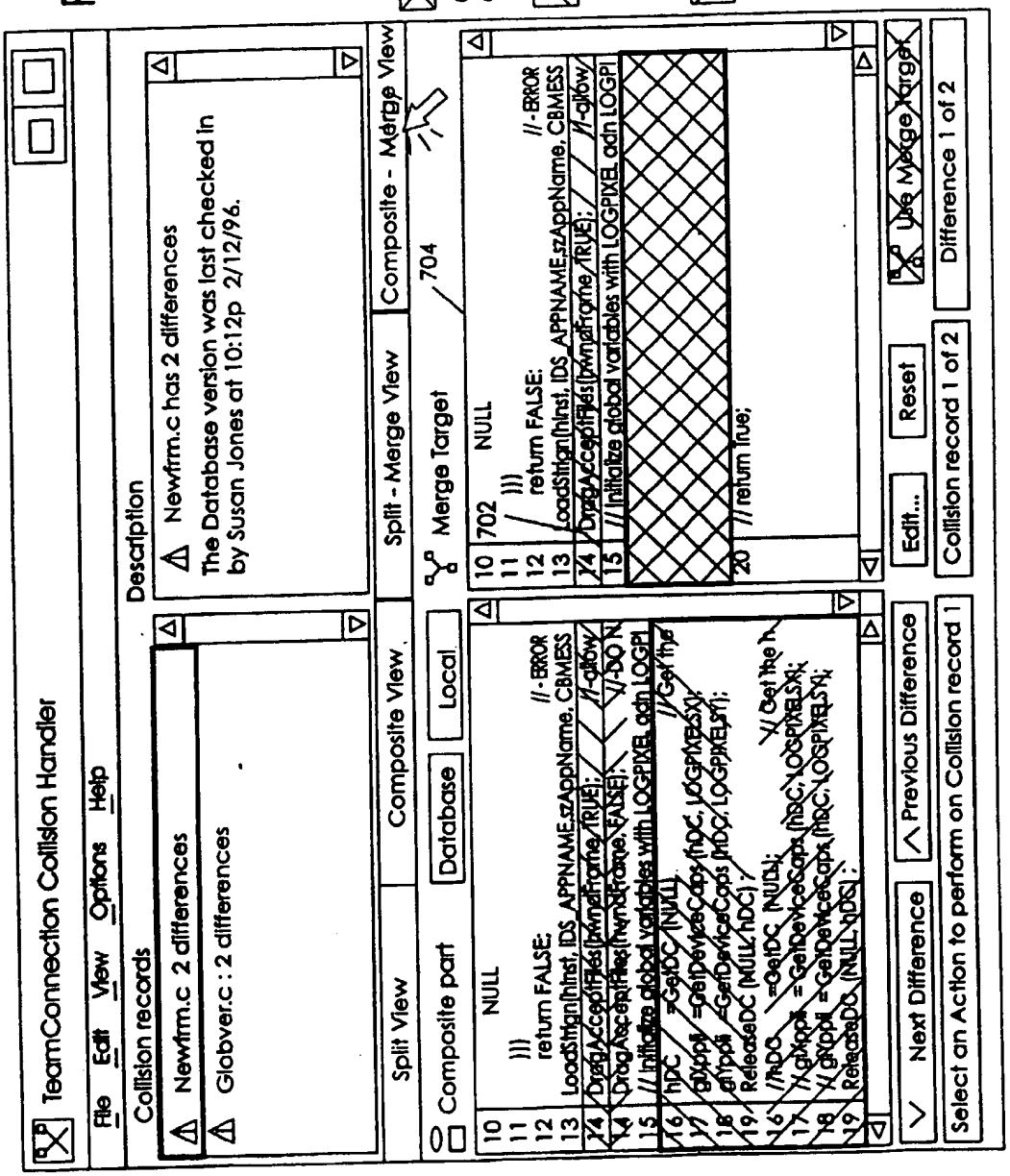

TeamConnection Collision Handler
File Edit View Options Help
Collision records
Newfrm.c 2 differences
Globver.c : 2 differences
Description
Newfrm.c has 2 differences
The Database version was last checked in by Susan Jones at 10:12p 2/12/96.
Split View
Composite View
Split - Merge View
Composite - Merge View
Composite part
Database
Local
Merge Target
704
NULL
return FALSE;
LoadStrgn(hInst, IDS_APPNAME,szAppName, CBMESS
DragAcceptFiles(hwndFrame,TRUE);
DragAcceptFiles(hwndFrame,FALSE);
// Initialize global variables with LOGPIXEL cdn LOGPI
GetDC (NULL);
GetDeviceCaps (hDC, LOGPIXELSX);
GetDeviceCaps (hDC, LOGPIXELSY);
ReleaseDC (NULL, hDC);
// Get the h
// -ERROR
// -dlow
// -DON
Next Difference
Previous Difference
Edit...
Reset
Use Merge Target
Collision record 1 of 2
Difference 1 of 2
Select an Action to perform on Collision record 1
GREYED OUT AREA
BLUE VERSION 1
RED VERSION 2

FIG. 8

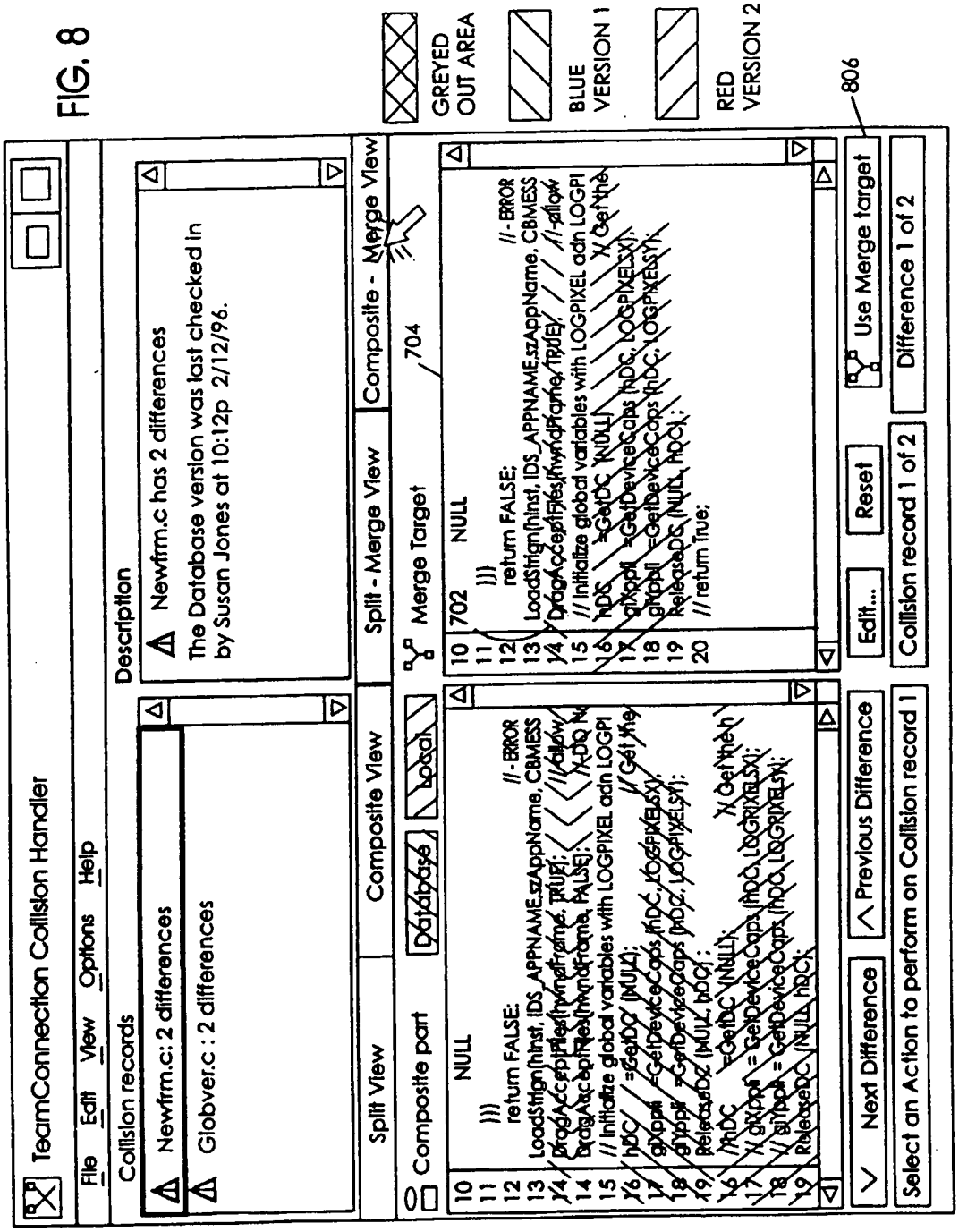

TeamConnection Collision Handler
File Edit View Options Help
Collision records
Newfrm.c: 2 differences
Globver.c : 2 differences
Description
Newfrm.c has 2 differences
The Database version was last checked in by Susan Jones at 10:12p 2/12/96.
Split View
Composite View
Split - Merge View
Composite - Merge View
Composite part
Database
Local
Merge Target
704
702
NULL
return FALSE;
Next Difference
Previous Difference
Edit...
Reset
Use Merge target
Collision record 1 of 2
Difference 1 of 2
Select an Action to perform on Collision record 1
806
GREYED OUT AREA
BLUE VERSION 1
RED VERSION 2

FIG. 9

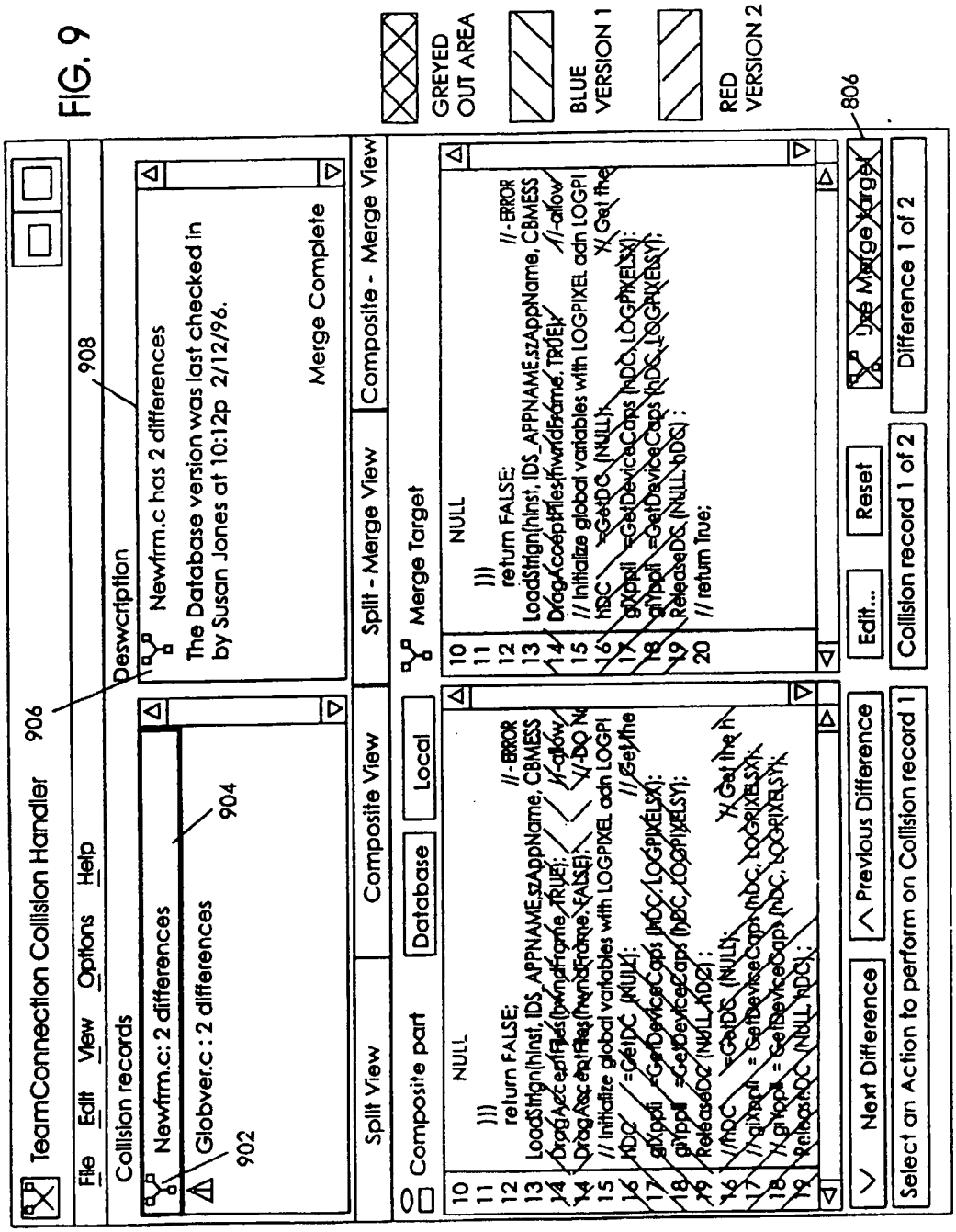

TeamConnection Collision Handler
906
File Edit View Options Help
908
Collision records
Deswcription
Newfrm.c: 2 differences
Globver.c : 2 differences
902
904
Newfrm.c has 2 differences
The Database version was last checked in by Susan Jones at 10:12p 2/12/96.
Merge Complete
Split View
Composite View
Split - Merge View
Composite - Merge View
Composite part
Database
Local
Merge Target
NULL
Next Difference
Previous Difference
Edit...
Reset
Use Merge Target
806
Select an Action to perform on Collision record 1
Collision record 1 of 2
Difference 1 of 2
GREYED OUT AREA
BLUE VERSION 1
RED VERSION 2

FIG. 10

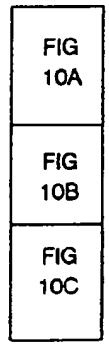
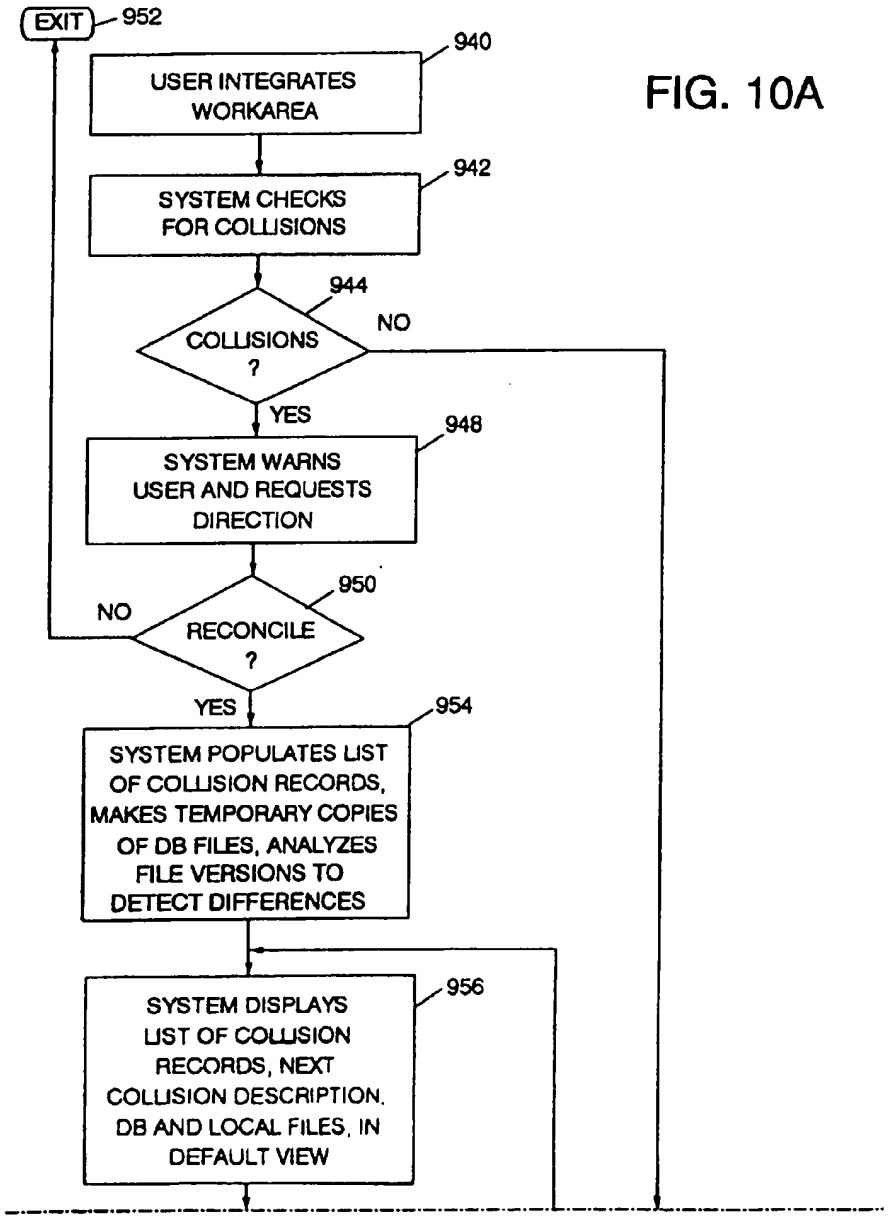


FIG. 10A



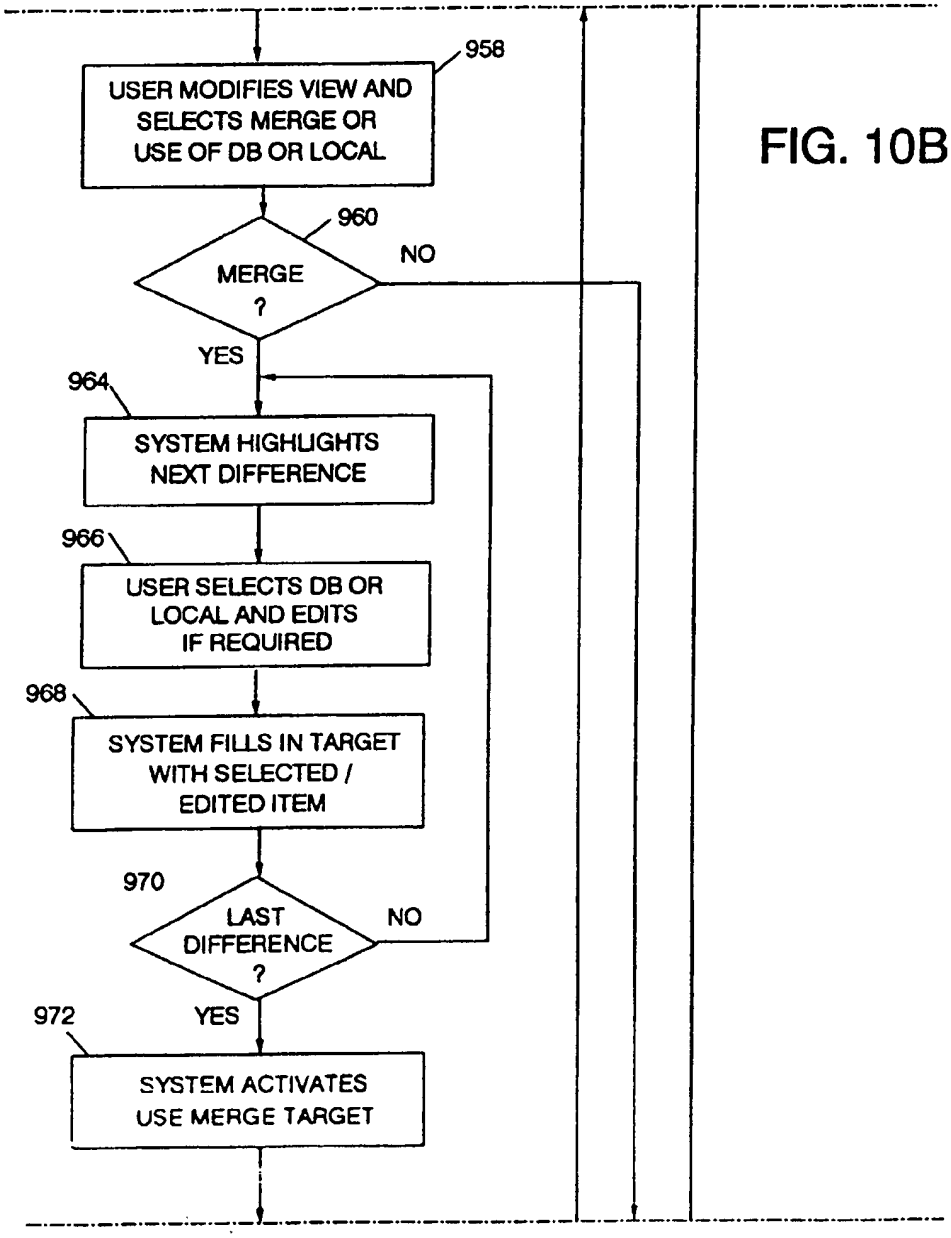

958
USER MODIFIES VIEW AND SELECTS MERGE OR USE OF DB OR LOCAL
960
MERGE ?
NO
YES
964
SYSTEM HIGHLIGHTS NEXT DIFFERENCE
966
USER SELECTS DB OR LOCAL AND EDITS IF REQUIRED
968
SYSTEM FILLS IN TARGET WITH SELECTED / EDITED ITEM
970
LAST DIFFERENCE ?
NO
YES
972
SYSTEM ACTIVATES USE MERGE TARGET

FIG. 10B

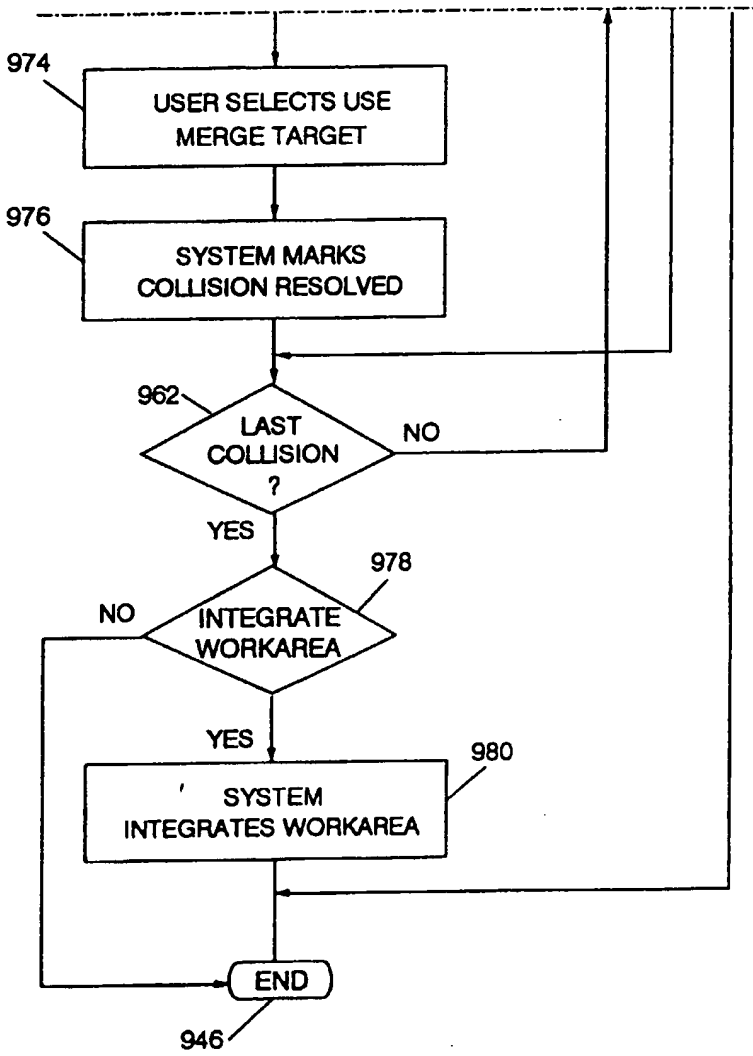

974
USER SELECTS USE MERGE TARGET
976
SYSTEM MARKS COLLISION RESOLVED
962
LAST COLLISION ?
NO
YES
978
INTEGRATE WORKAREA
NO
YES
980
SYSTEM INTEGRATES WORKAREA
END
946

FIG. 10C

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 907 125 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
25.01.2006 Bulletin 2006/04

(43) Date of publication A2:
07.04.1999 Bulletin 1999/14

(21) Application number: **98306288.6**

(22) Date of filing: **06.08.1998**

(51) Int Cl.:
G06F 9/44 (2006.01) *G06F 17/30* (2006.01)

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **05.09.1997 US 923910**

(71) Applicant: **International Business Machines Corporation**
Armonk, NY 10504 (US)

(72) Inventors:
- **Percival, Lynn Cleveland**
  Raleigh, North Carolina 27612 (US)
- **Scanlon, John Lane**
  Raleigh, North Carolina 27613 (US)

(74) Representative: **Zerbi, Guido Maria**
Intellectual Property Department,
IBM United Kingdom Ltd.,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Method and system for automatically detecting and selecting updated versions of a set of files

(57) A list of various versions of file sets that need to be reconciled (collisions) are automatically loaded into a collision handler. The user works through the list, resolving differences by selecting one of the files or by merging several of them through selection and modification, as required. The user is provided with buttons that allow rapid changes between different styles of views (Split, Composite, Split-Merge, Composite-Merge) which allows the user to see the files in a number of different ways to further facilitate the process.

EP 0 907 125 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 98 30 6288

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	MUNSON ET AL: "A flexible object merging framework" CSCW. PROCEEDINGS OF THE CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK, 22 October 1994 (1994-10-22), pages 231-242, XP002074488 * the whole document *	1-15	G06F9/44 G06F17/30
X	TERRY D N ET AL: "MANGING UPDATE CONFLICTS IN BAYOU, A WEAKLY CONNECTED REPLICATED STORAGE SYSTEM" OPERATING SYSTEMS REVIEW, ACM PRESS, vol. 29, no. 5, 1 December 1995 (1995-12-01), pages 172-183, XP000565907 * the whole document *	1-15	
X	KISTLER J J ET AL: "DISCONNECTED OPERATION IN THE CODA FILE SYSTEM" ACM TRANSACTIONS ON COMPUTER SYSTEMS, ACM, NEW YORK, NY, US, vol. 10, no. 1, 1 February 1992 (1992-02-01), pages 3-25, XP000323223 ISSN: 0734-2071 * the whole document *	1-15	

TECHNICAL FIELDS SEARCHED (IPC)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	29 November 2005	Wienold, N

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)